Software Engineering

lecture Resource

https://ocw.mit.edu/

https://hbr.org/1963/09/the-abcs-of-the-critical-path-method

Lecture 1 – Critical Path Methods (CPM)

Eng. Sally Jarkas

Eng Aya Joumaa

CPM Assumptions

- Project consists of a collection of well defined tasks (jobs)
- Project ends when all jobs completed
- Jobs may be started and stopped independently of each other within a given sequence
- Jobs are ordered

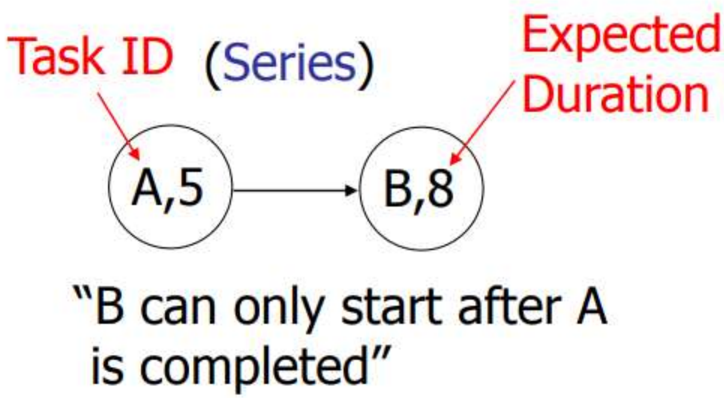
Project Graph

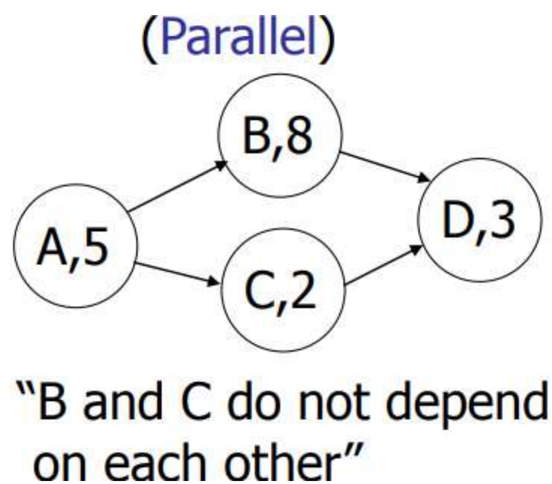
- Each job is drawn on a graph as a circle*
- Connect each job with immediate predecessor(s) – unidirectional arrows "->"
- Jobs with no predecessor connect to "Start"
- Jobs with no successors connect to "Finish"
- "Start" and "Finish" are pseudo-jobs of length 0
- Total time of each path is the sum of job times
- Path with the longest total time "critical path"
- There can be multiple critical paths
- minimum time to complete project

* or other symbol, see before

CPM 101

- Represent a project (set of task) as a network using graph theory
  - Capture task durations
  - Capture task logic (dependencies)





Critical Path

- CP is the “bottleneck route”
- Shortening or lengthening tasks on the critical path directly affects project finish
- Duration of “non-critical” tasks is irrelevant
- “Crashing” all jobs is ineffective, focus on the few % of jobs that are on the CP
- Previously non-critical tasks can become critical
- Lengthening of non-critical tasks can also shift the critical path

Some Network Definitions

- All **activities** on the **critical path** have **zero slack**
- **Slack** defines how long **non-critical activities** can be **delayed without delaying the project**
- **Slack** = the activity's **late finish minus its early finish** (or its **late start minus its early start**)
- Earliest Start (**ES**) = the earliest finish of the immediately preceding activity
- Earliest Finish (**EF**) = is the **ES plus** the **activity time**
- Latest Start (**LS**) and Latest Finish (**LF**) = the latest an activity can start (LS) or finish (LF) without delaying the project completion

Critical Path Algorithm

- For large projects there are many paths
- Need a algorithm to identify the CP efficiently
- Develop information about each task in context of the overall project
- Times
  - start time (S)
  - For each job: Earliest Start (ES)
  - Earliest start time of a job if all its predecessors start at ES
  - Job duration: t
  - Earliest Finish (EF)=(ES)+t
  - Finish time (F) – earliest finish time of the overall project
- Show algorithm using project graph

CP Algorithm

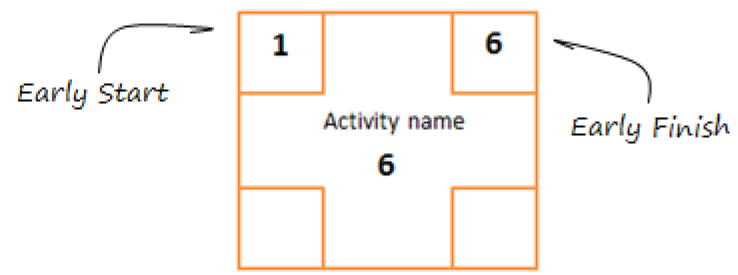
1. Mark the value of S to left and right of Start
2. Consider any new unmarked job, all of whose predecessors have been marked. Mark to the left of the new job the largest number to the right of its immediate predecessors: (ES)
3. Add to ES the job time t and mark result to the right (EF)
4. Stop when Finish has been reached

Critical path method

- A *forward pass* performs schedule calculations that identify the early start and finish dates of tasks and the project
- A *backward pass* performs schedule calculations that identify the late start and finish dates of tasks and the project, as well as total and free float

Critical path method *forward pass*

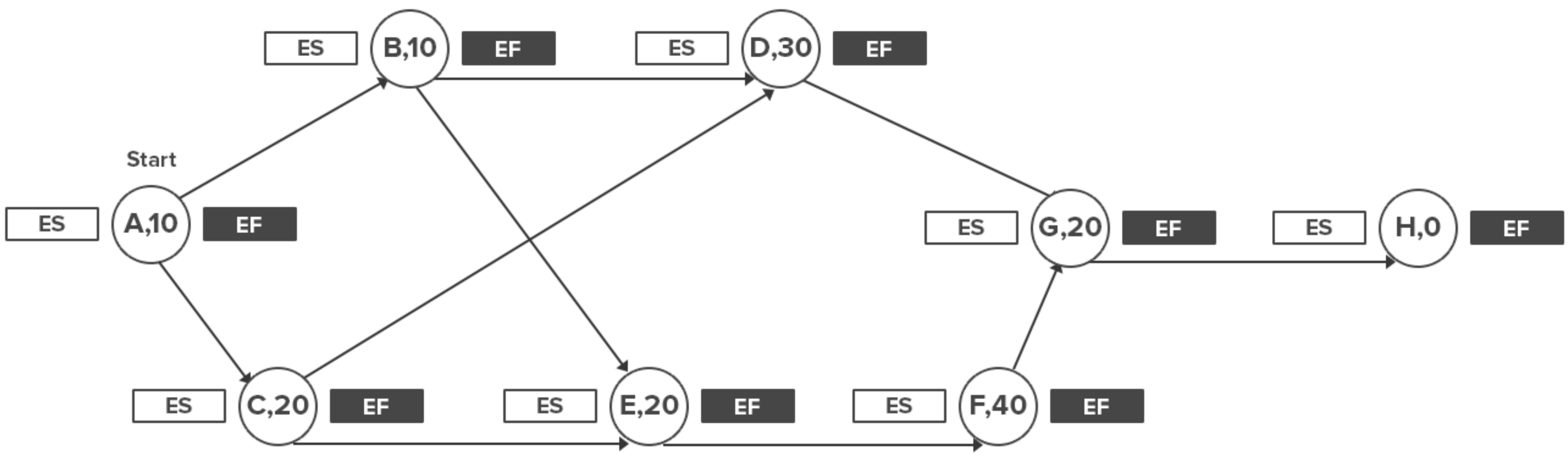
- A *forward pass analysis* performs schedule calculations that identify the early start and finish dates of activities and the project
  - *Early Start Date* (ES). ES represents the theoretically earliest date a activity can start
    - ES = Maximum EF of predecessor activity(-ies)
  - *Early Finish Date* (EF). EF represents the theoretically earliest date a activity can finish
    - EF = ES + duration of activity



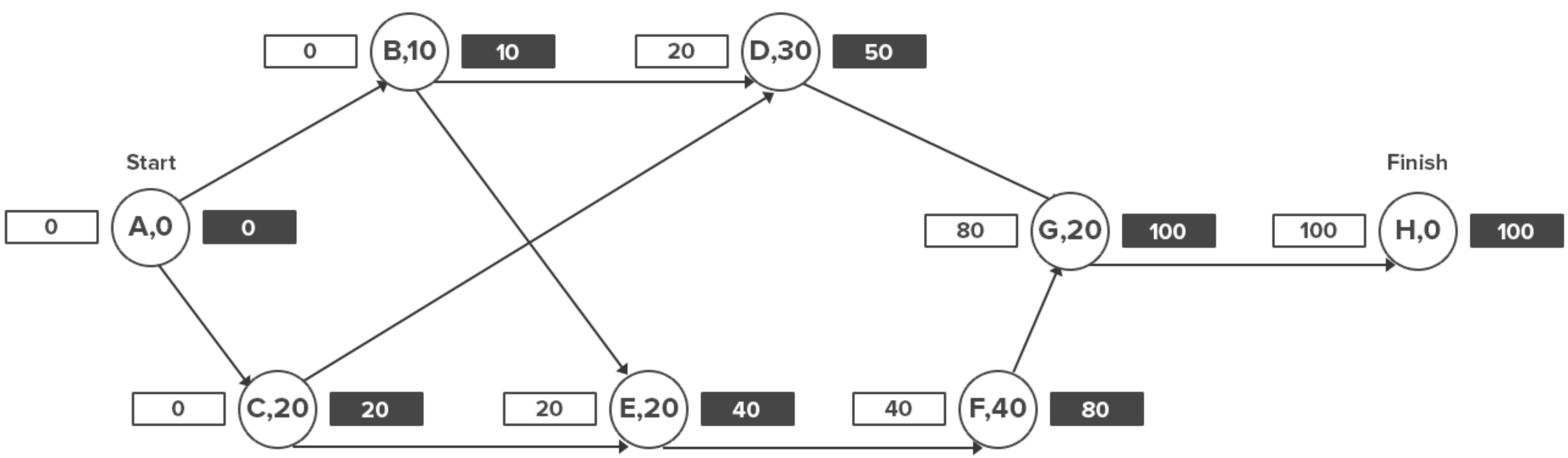
Simple Example: Job List

Job #	Description	Immediate Predecessors	Time [min]
A	Start		0
B	Get materials for X	A	10
C	Get materials for Y	A	20
D	Turn X on lathe	B,C	30
E	Turn Y on lathe	B,C	20
F	Polish Y	E	40
G	Assemble X and Y	D,F	20
H	Finish	G	0

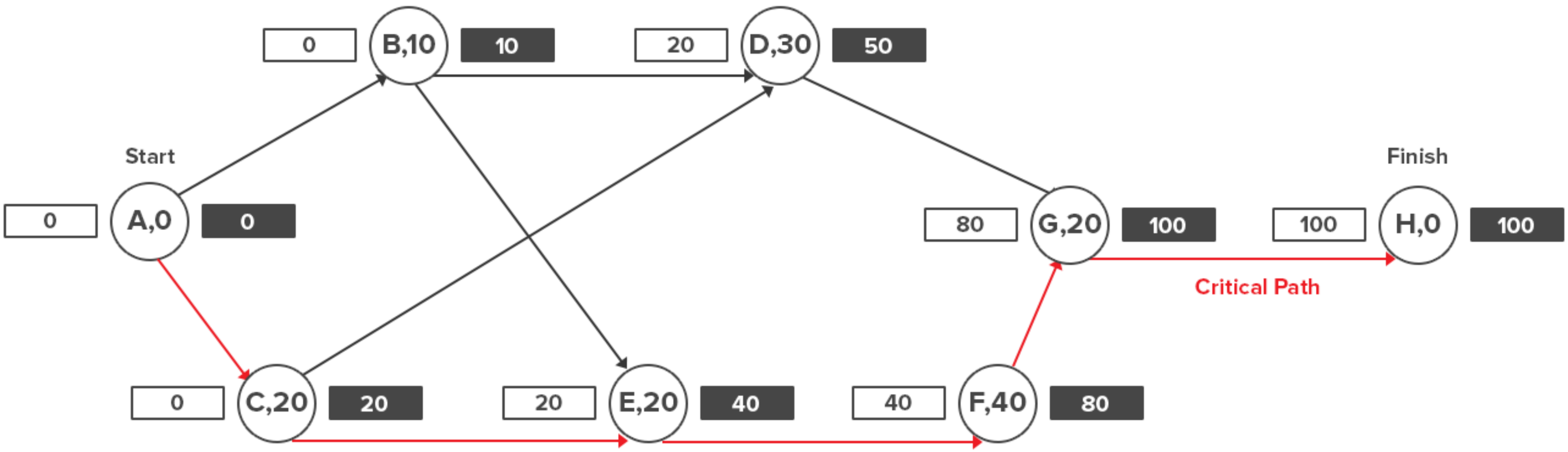
CP Algorithm – ES EF



CP Algorithm – ES EF

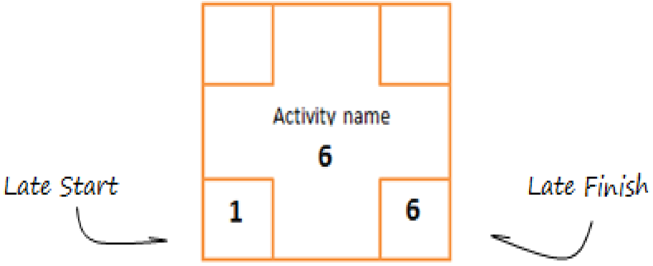


CP Algorithm – ES EF

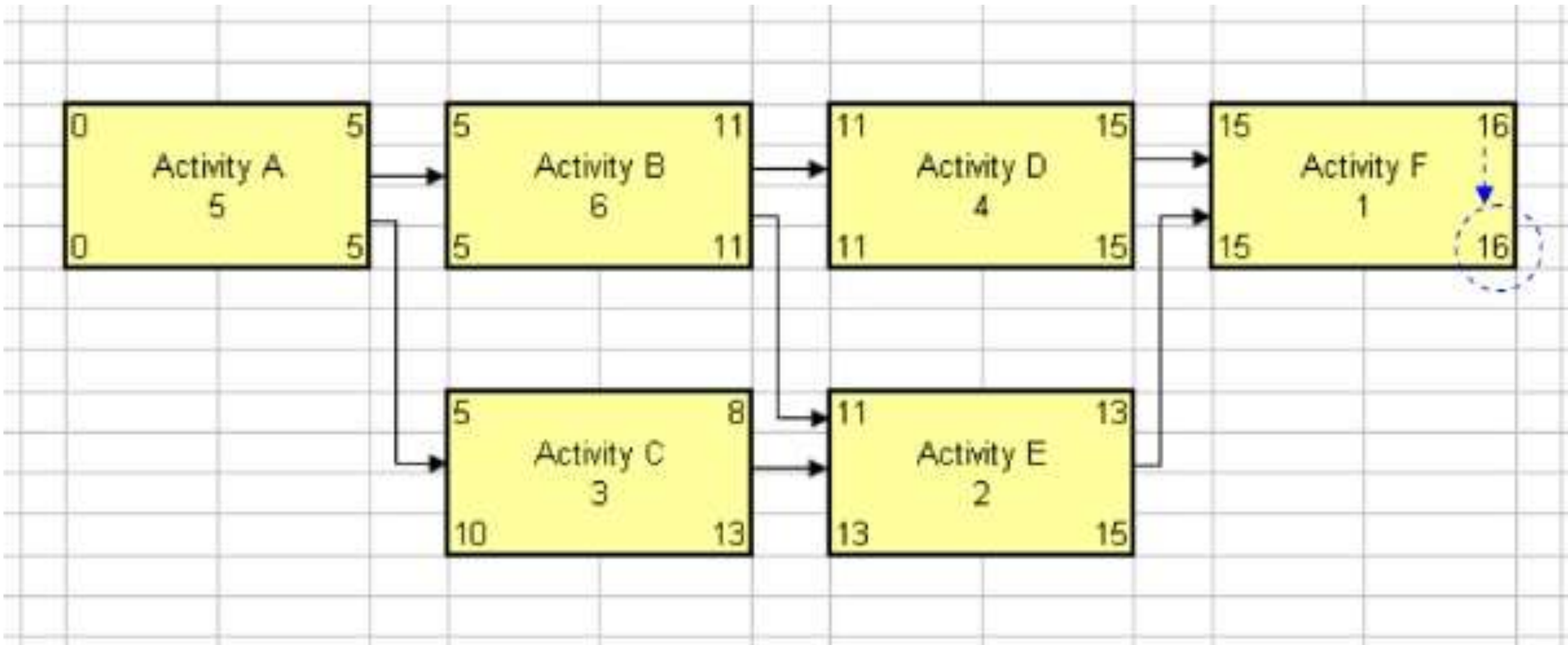


Critical path method - *backward pass*

- A *backward pass analysis* performs schedule calculations that identify the late start and finish dates of activities and the project, as well as total and free float
  - *Late Start Date* (LS). LS represents the theoretically latest date a activity can start without delaying the project
    - **LS = LF – duration of activity**
  - *Late Finish Date* (LF). LF represents the theoretically latest date a activity can finish without delaying the project
    - **LF = Minimum LS of successor activity(-i**



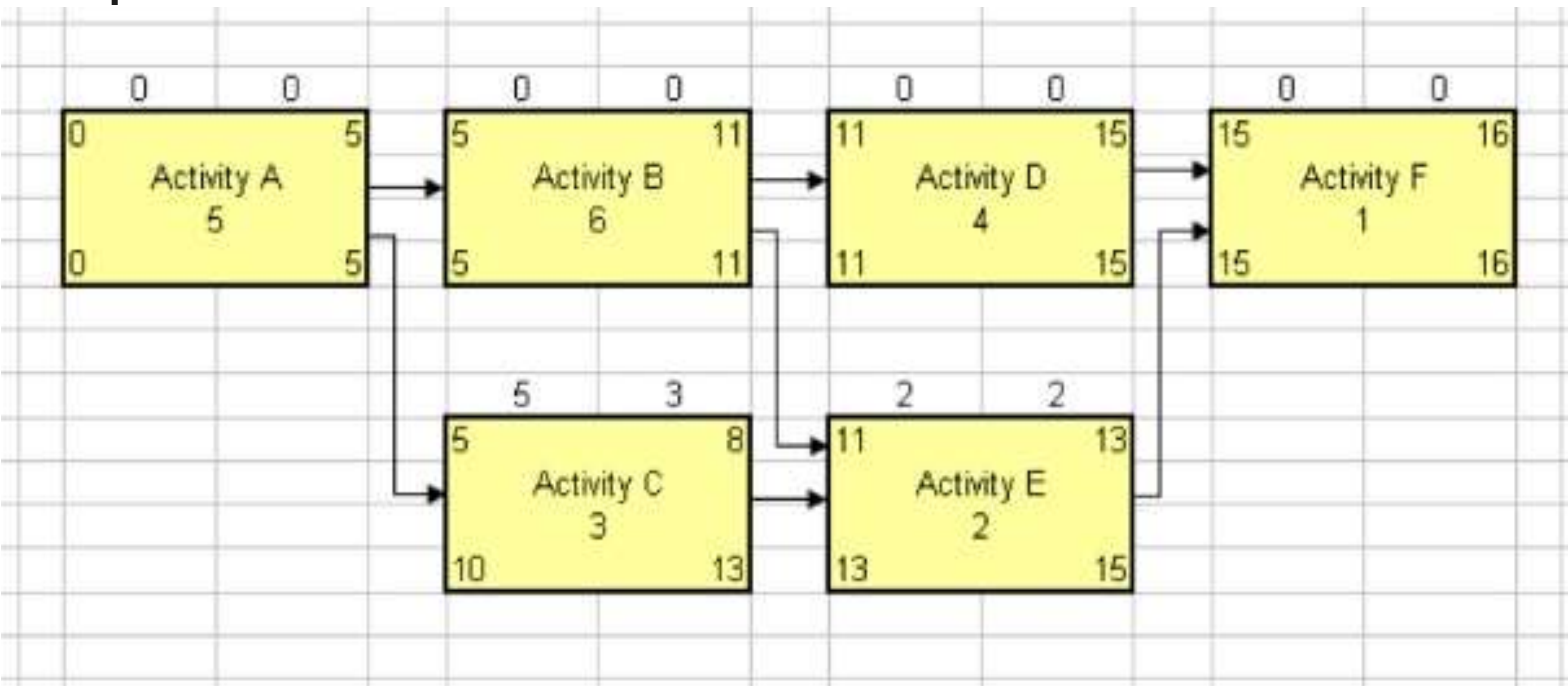
CP Algorithm – LS LF



CP Algorithm – Total , Free Slack

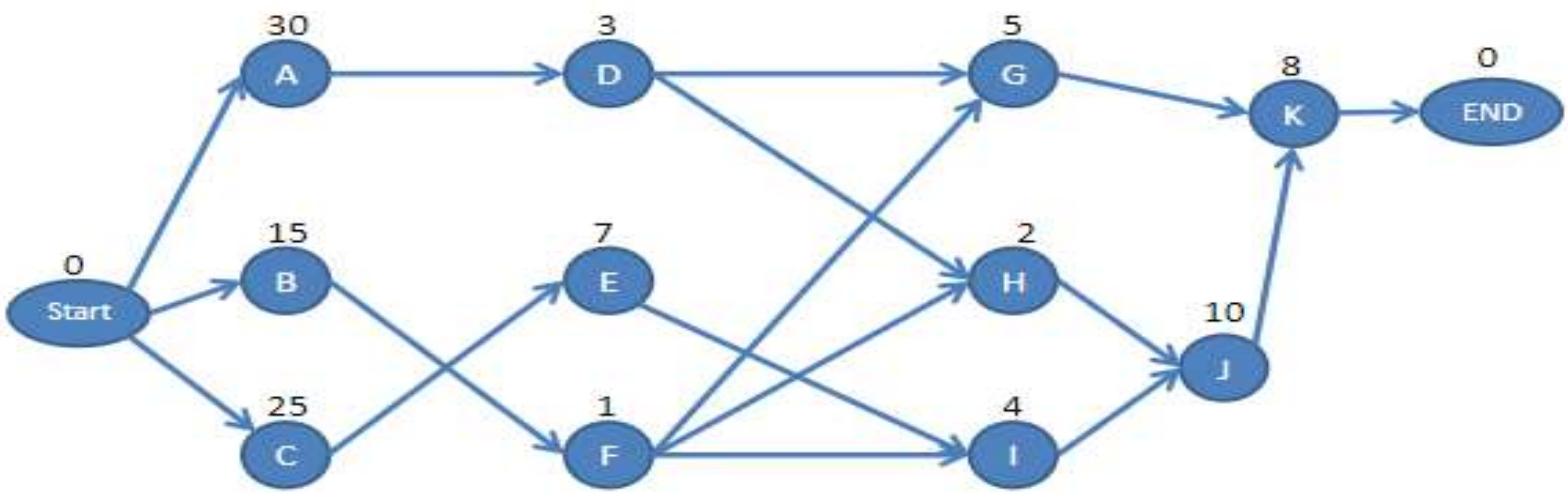
- Total Float = LS – ES (it is also calculated by LF – EF)
- Free Float = Lowest ES of successors – EF

CP Algorithm – Total , Free Slack



Activity	Immediate Predecessors	Time estimate (days)
A	-	30
B	-	15
C	-	25
D	A	3
E	C	7
F	B	1
G	D, F	5
H	D, F	2
I	E, F	4
J	H, I	10
K	G, J	8

Activity on Node network

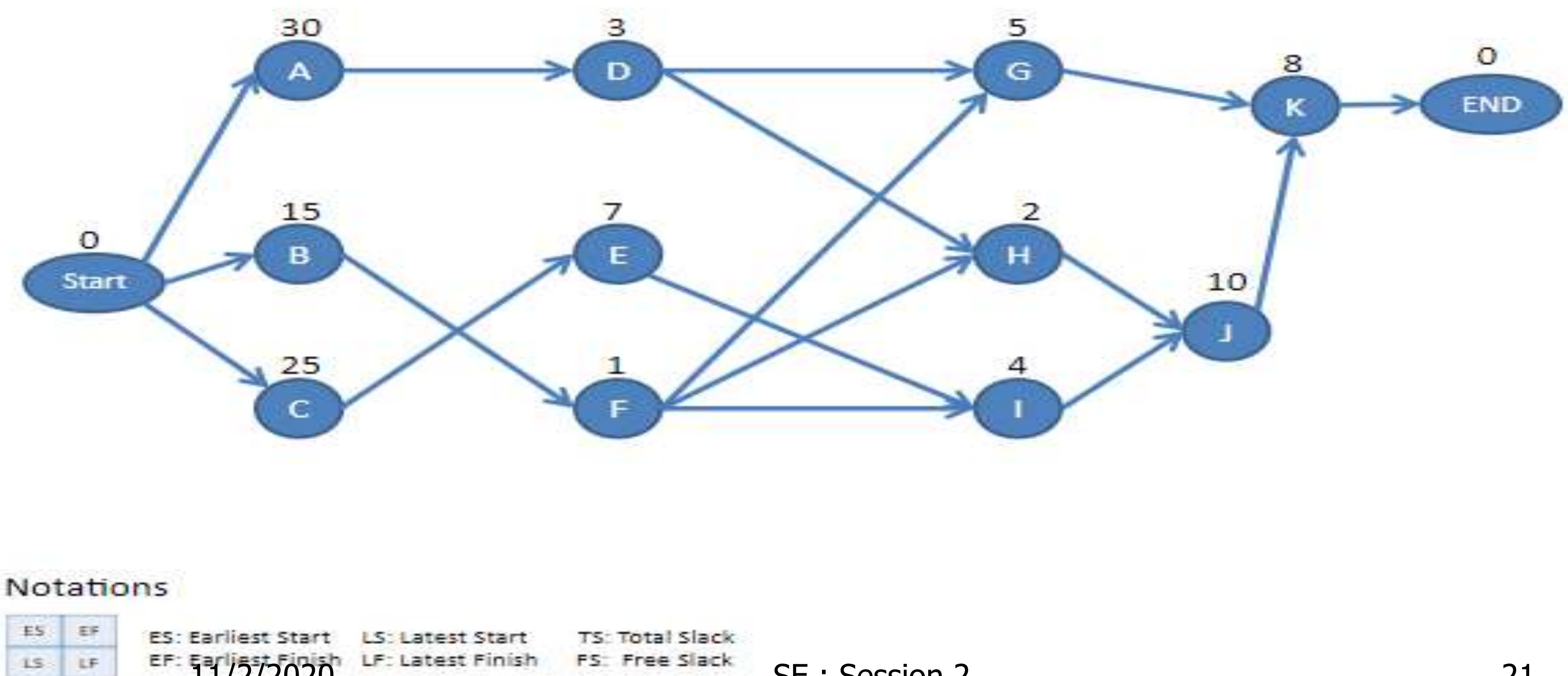


Notations

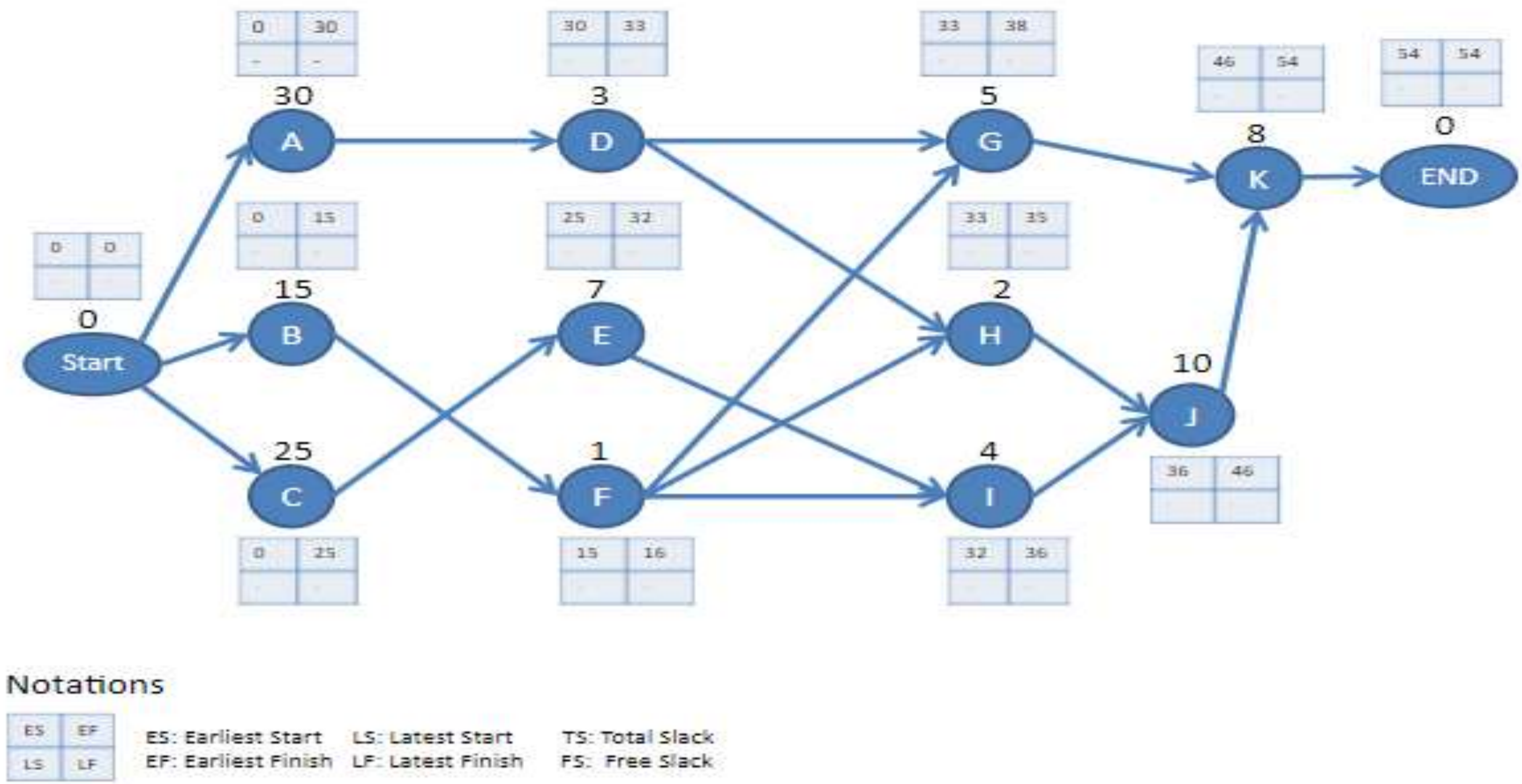
ES	EF
LS	LF

ES: Earliest Start
EF: Earliest Finish
LS: Latest Start
LF: Latest Finish
TS: Total Slack
FS: Free Slack

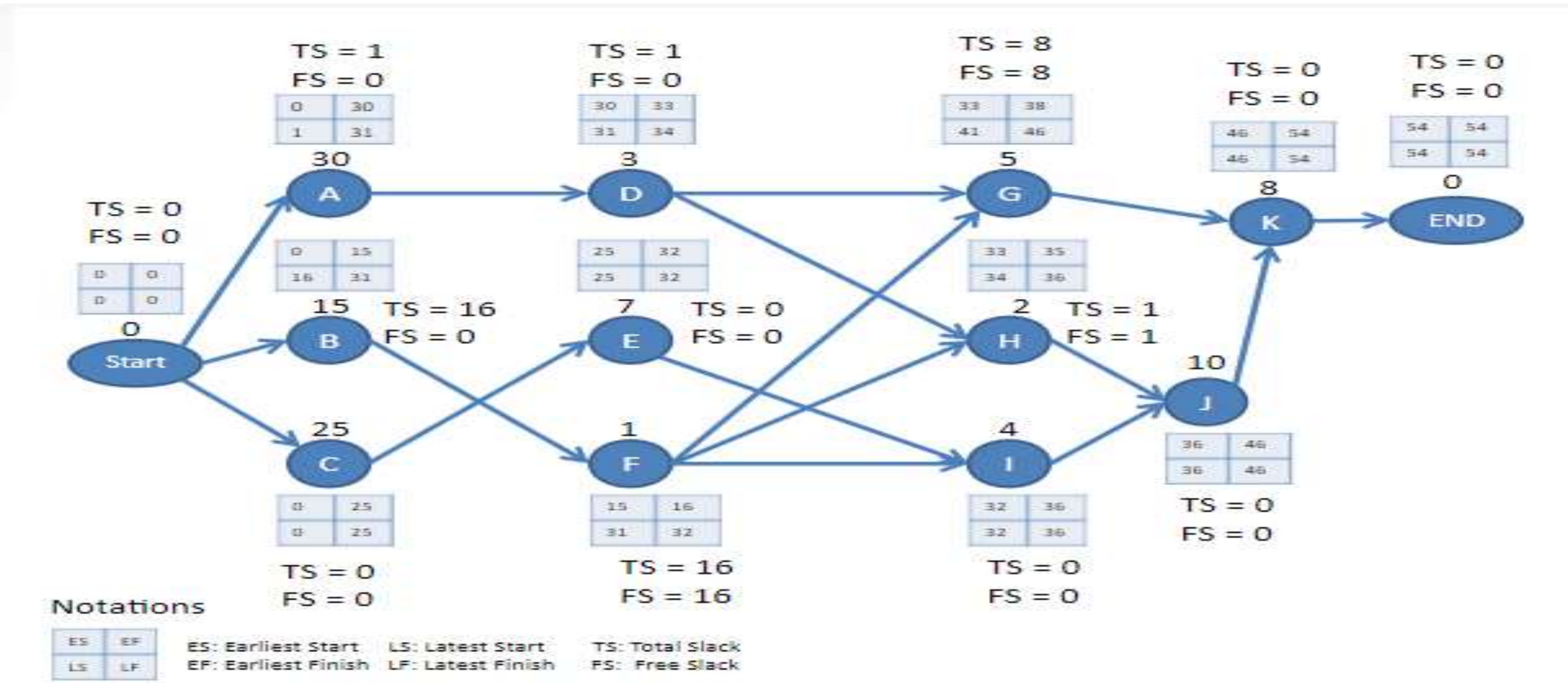
Earliest Start and Earliest Finish



Latest Start and Latest Finish

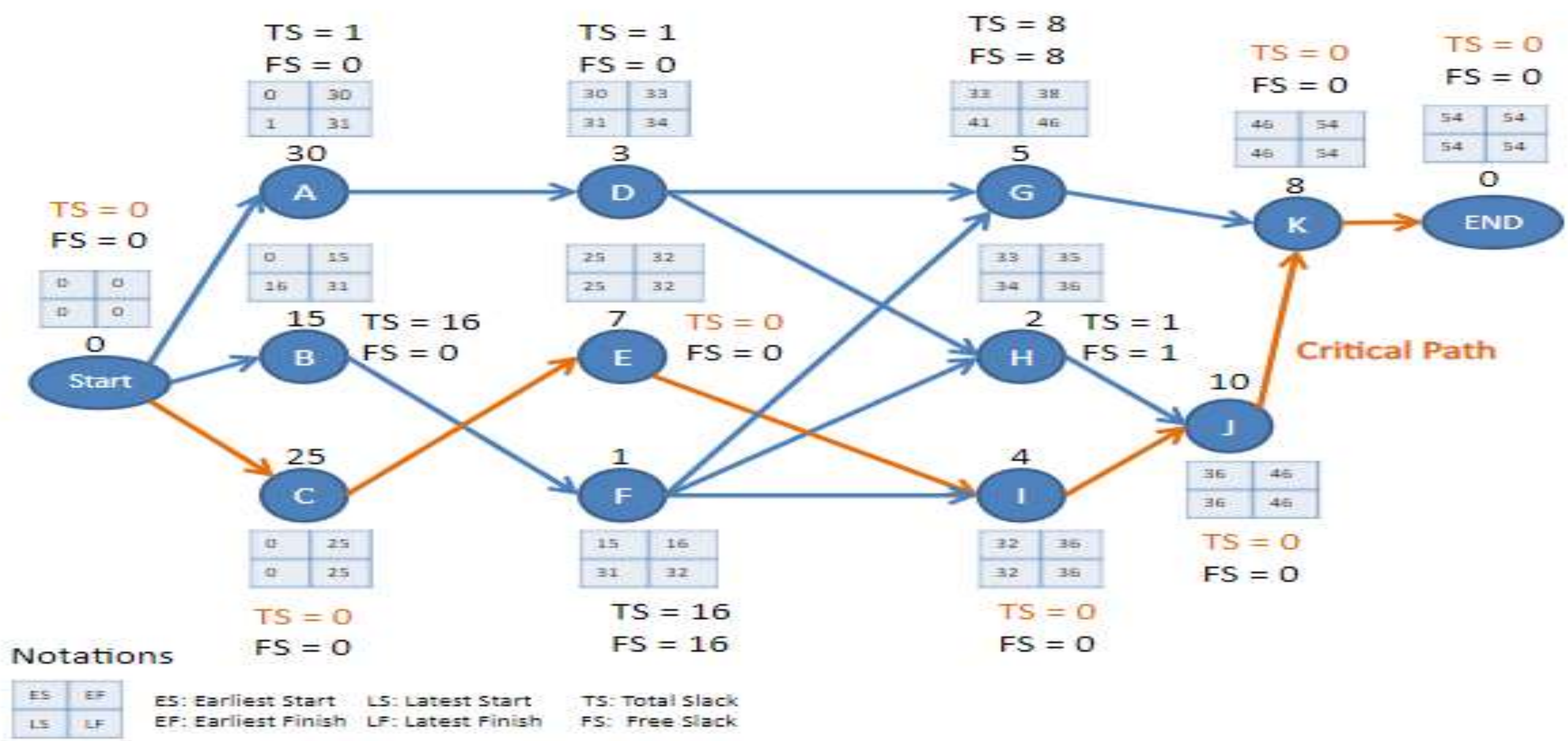


TS , FS



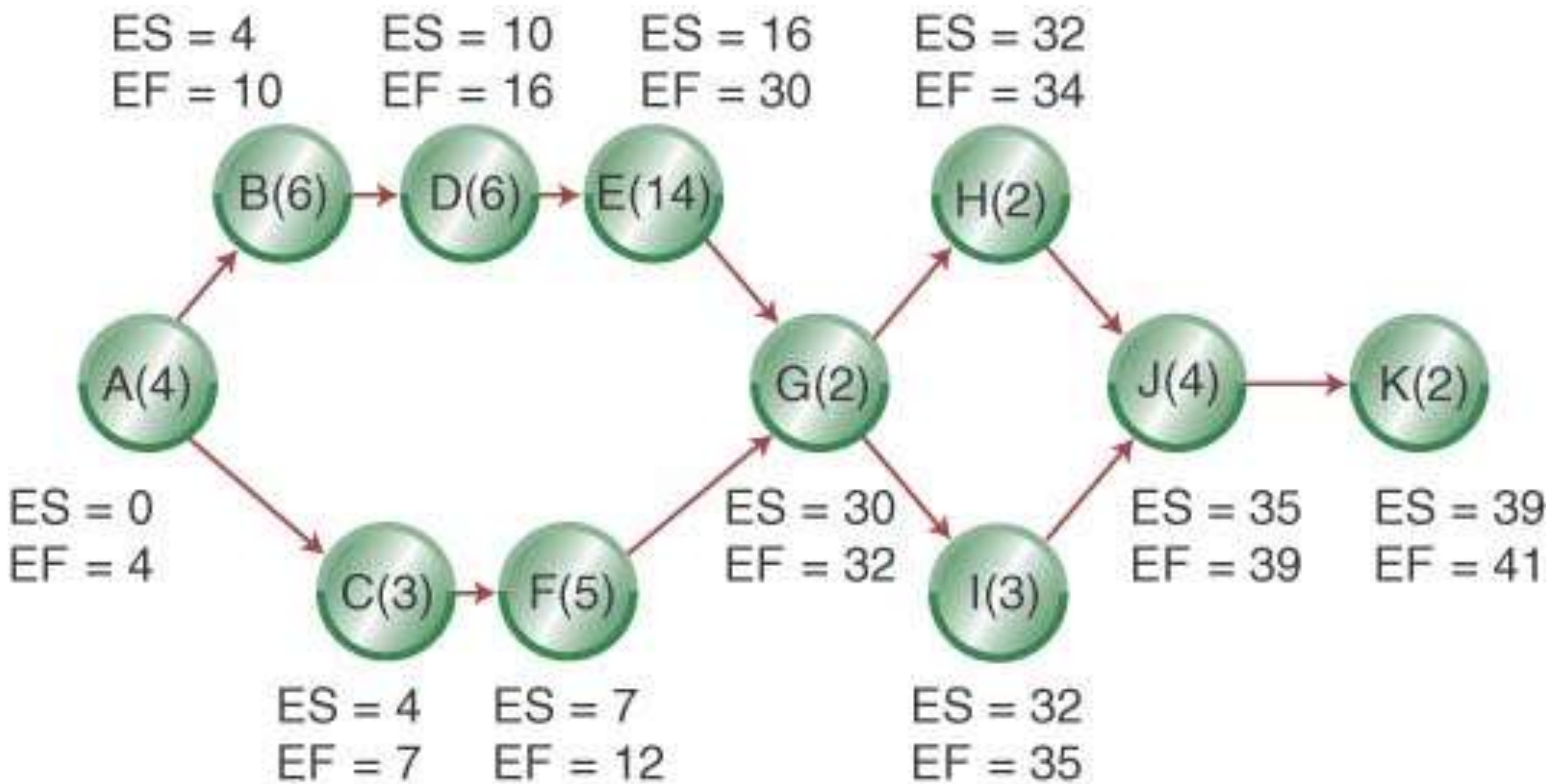
TS = 1
FS = 0
0 30
1 31
30
A
TS = 1
FS = 0
30 33
31 34
3
D
TS = 8
FS = 8
33 38
41 46
5
G
TS = 0
FS = 0
46 54
46 54
8
K
TS = 0
FS = 0
54 54
54 54
0
END
TS = 0
FS = 0
0 0
0 0
0
Start
0 15
16 31
15
B
TS = 16
FS = 0
25 32
25 32
7
E
TS = 0
FS = 0
33 35
34 36
2
H
TS = 1
FS = 1
10
J
36 46
36 46
TS = 0
FS = 0
25
C
0 25
0 25
TS = 0
FS = 0
1
F
15 16
31 32
TS = 16
FS = 16
4
I
32 36
32 36
TS = 0
FS = 0
Notations
ES EF
LS LF
ES: Earliest Start
EF: Earliest Finish
LS: Latest Start
LF: Latest Finish
TS: Total Slack
FS: Free Slack

Critical Path



Task	Predecessor	Duration
A		4
B	A	6
C	A	3
D	B	6
E	D	14
F	C	5
G	E,F	2
H	G	2
I	G	3
J	H,I	4
K	J	2

ES, EF Network



LS, LF Network